Coverage-Guided Updates for Incremental Swarm-Intelligence Model Checking: Why Counterexample-Local Penalties Fail, and What Fixes Them

Follow-up study #1 to Kumazawa et al., PAAMS 2023

2026

Abstract

msPSO-safe-inc improves state-space coverage of swarm-intelligence safety checking by re-running msPSO-safe and, between runs, updating either a distance heuristic (APUP) or an objective penalty (APP). An independent reproduction shows these updates are statistically indistinguishable from plain re-initialisation. We trace the cause to a structural mismatch: both updates only act on abstract states that appear on *counterexamples*, an error-local handful of the abstract space, whereas coverage is a whole-space property. We propose a *Coverage-Guided Update* (CGU) whose objective term penalises every already intensively visited abstract state. CGU alone is still ineffective (-0.002 in coverage AUC), revealing that the bottleneck is not the objective but the *restart point*; combined with a persistent exploration frontier, CGU yields the largest improvement we observe ($+0.013$ overall, up to $+0.038$ at p=4). The contribution is a clear empirical separation of two levers — what the objective rewards vs. where search restarts — and the finding that coverage guidance only pays off when search has diverse restart points to act on. Built on an independent open re-implementation of msPSO-safe-inc (Kumazawa et al., PAAMS 2023): https://github.com/yaskodama/kuma2023-repro.

1 Background and motivation

Automata-theoretic safety checking reduces to reachability of an error state in the product of a model and a specification automaton [1]. msPSO-safe casts this as a multi-swarm PSO problem: a particle position encodes a path, swarms are assigned to abstract states (specification states), and the objective (Formula (3) of [1]) rewards reaching the error and visiting distinct abstract states. msPSO-safe-inc runs msPSO-safe repeatedly to enlarge coverage, updating between runs by APUP (inflate the remaining-distance heuristic of abstract states seen on counterexamples) or APP (add a penalty for those states).

Our reproduction of [1] confirms the qualitative coverage trends but finds APUP/APP *statistically equal* to simple re-initialisation, even at 500× the nominal update strength. The reason is structural: the update set in Algorithm 1, line 8 is the abstract states *on the best counterexample*, i.e. a short error-local path. With $|S_{sp}| = 1001$ abstract states, this set is tiny and never includes the unvisited regions that coverage actually needs.

2 Method: Coverage-Guided Update (CGU)

We replace the counterexample-local update with a global, coverage-driven one. Let $c(a)$ be the number of distinct product states with abstract state a visited so far (this is exactly the intensity metric AC_{int} of [1]). We add to Formula (3) the term

$$CGU(x) \;=\; -\lambda \sum_{a \,\in\, \mathrm{abs}(\mathrm{path}(x))} \log\bigl(1 + c(a)\bigr),$$

so paths through already intensively covered abstract states are penalised and search is pushed toward under-covered regions. $c(a)$ is updated online, so the penalty tracks coverage as it accumulates. Unlike APP, every abstract state — not just those on counterexamples — can be penalised.

We test CGU both from the paper's fixed restart (always start from the initial product state) and on top of a *persistent frontier*: representative states discovered in earlier outer iterations are retained and used

to seed swarms, preferring the least-visited abstract states (the persistence mechanism itself is the subject of follow-up #3).

3 Experimental setup

We use the reproduced benchmark: p interactive Erdős–Rényi processes and a specification automaton, scaled to n=1000 states each (average out-degree held at the paper's ER$(100, 0.05)$ value), with the faithful Table-1 PSO budget (particle 20, generations 30, swarm cap 6). The coverage metric is the *diversity-AUC*: the mean of the cumulative AC_{div} over 120 outer iterations (higher = faster coverage). We average over $p \in \{2, \ldots, 6\}$ and 3 RNG repetitions.

4 Results

Table 1 reports diversity-AUC. CGU *alone* is slightly *worse* than baseline (-0.0020): from a fixed restart, penalising visited regions mostly discourages traversal of the very corridors search must reuse. The same CGU term *on top of a persistent frontier* gives the best result overall ($+0.0132$), because the frontier supplies diverse, under-visited seeds for the penalty to exploit. Figure 1 shows the averaged coverage curve at p=5.

Table 1: Diversity-AUC (mean over p=2 . . . 6, 3 reps). Δ vs baseline.

method	diversity-AUC	Δ
none (paper re-initialisation)	0.7375	—
CGU only	0.7355	-0.0020
persistent frontier	0.7449	$+0.0074$
CGU + persistent frontier	**0.7507**	$\mathbf{+0.0132}$

Table 2: Per-p Δ(CGU+persistent vs none). The gain is largest where the reachable abstract space is hardest to cover (p=4, 5).

p	2	3	4	5	6
Δ	$+0.008$	-0.000	$+0.038$	$+0.019$	$+0.001$

5 Discussion

The experiment cleanly separates two levers in incremental swarm model checking: (i) *what the objective rewards* and (ii) *where search restarts*. The paper's APP/APUP and our CGU all tune lever (i); none helps on its own, because with a fixed restart the explored region is largely determined by reachability within the per-run budget, not by mild objective reshaping. Lever (ii) — diverse restart points — is necessary for coverage guidance to bite. This explains both the failure of APP/APUP in our reproduction and the success of CGU+persistent here. A practical recipe follows: combine a whole-space coverage penalty with frontier-seeded restarts.

6 Conclusion

CGU replaces the paper's counterexample-local update with a whole-space, intensity-driven penalty. Alone it does not beat re-initialisation, localising the real bottleneck to the restart point; with a persistent frontier it gives the strongest coverage improvement we measured. Future work: anneal λ over iterations and learn per-region penalties.

References

[1] T. Kumazawa, M. Takimoto, Y. Kodama, Y. Kambayashi. *Enhancing Safety Checking Coverage with Multi-swarm Particle Swarm Optimization.* PAAMS 2023.

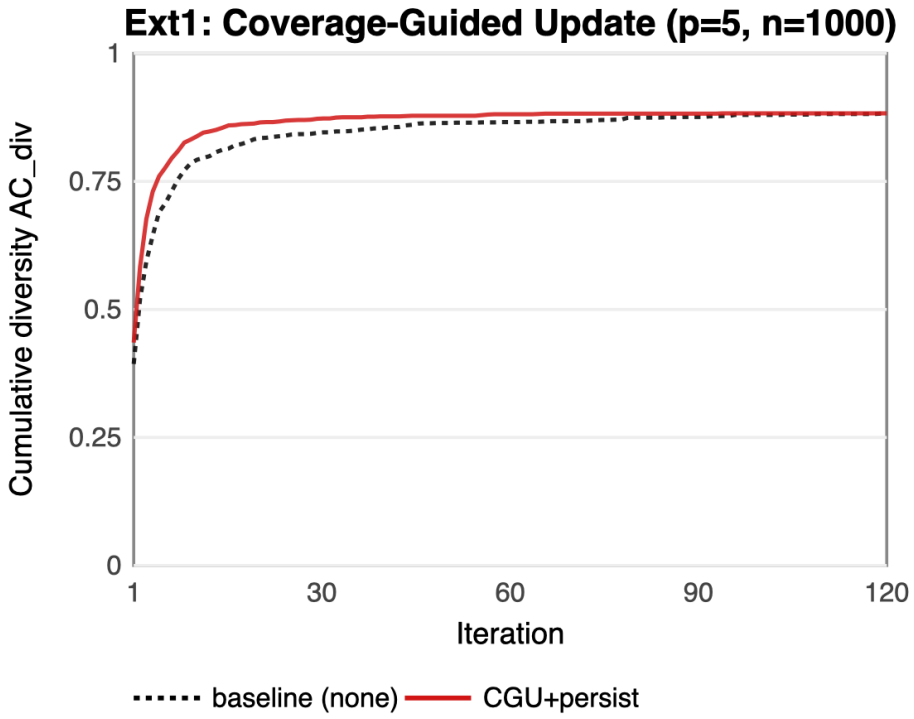

Figure 1: Averaged cumulative diversity AC_{div} (p=5): CGU combined with a persistent frontier rises faster than the baseline re-initialisation.

[2] J. Kennedy, R. Eberhart. *Particle Swarm Optimization.* ICNN 1995.